May 2014

Smartphone Radio Application

Edison Jimenez
Worcester Polytechnic Institute

Seth Martin Crampton
Worcester Polytechnic Institute

WPI

Smartphone Radio Application

A Major Qualifying Project Report
Submitted to the Faculty of
Worcester Polytechnic Institute

In partial fulfillment of the requirements for the
Degree of Bachelor of Science

Submitted on May 6th, 2014

Written by
Edison Jimenez
Seth Crampton

Approved by
Professor Emmanuel Agu
Professor Scott Barton

Table of Contents

List of Figures

Abstract

WICN (90.5 FM) is a public jazz radio station located in Worcester, MA that broadcasts National Public Radio (NPR) news along with Jazz, Blues, Soul, Bluegrass, Folk, Americana, and Latin Jazz music to an audience of over forty thousand. During the 2013-14 academic year, two GQP groups from Worcester Polytechnic Institute conducted market research on the benefit of developing a smartphone application that would allow the station's listeners to access the music stream along with other listener requested features. This research concluded that a mobile application would be highly beneficial. Therefore, the goal of this project was to develop a native mobile application for the Android Operating System that would feature live radio streaming capabilities, display parts of the station's website in a mobile-friendly format, alarm functions, and donation capabilities.

Chapter 1: Introduction

WICN is a public radio station based in Worcester that offers 'Jazz+' music programming and National Public Radio (NPR) news. The station currently broadcasts via the air (in standard and High Definition formats) and online via NPR Online Services. In order to reach a larger demographic and potentially increase revenue, the station recently became interested in developing a mobile application that would enable their listeners to listen to the online broadcast and access additional content pertaining to programming and local performances. To gauge how an application would mutually benefit the station and its listeners, WICN collaborated with graduate students from the Worcester Polytechnic Institute School of Business to perform market research. Part of this research consisted of surveys that asked listeners what features they would want in a mobile application.

Our project builds on the conclusions of this market research, which recommended live radio streaming capabilities, displaying parts of the station's website in a mobile-friendly format, alarm functions, and donation capabilities. This project is also based on communications between our project team and WICN's staff members. Our project group spent the Spring semester of the 2013-2014 academic school year developing the Android version of this application. This document will comprehensively describe our entire development process including details ranging from initial conversations to the final product we delivered to the station.

Chapter 2: Background and Related Work

2.1 Analysis of GQP Reports

Part of this project consisted of reviewing two GQP research reports in order to attain a clearer understanding as to what a potential mobile application should contain. The first report titled *WICN Mobile Application Market Research*, contained most of the information we needed that directly correlated with our application development including listener preferences, demographics, etc. The second report titled *Strategic Marketing Plan for Collaboration Between WICN Public Radio and WPI* contained much more demographic information.

2.1.1 GQP #105 - WICN Mobile Application Market Research

This GQP report (Diana, N.; Yao-Hua Chung, K.; & Li, X. 2013), specifically focused on how viable a mobile application would benefit the WICN radio station. This research was helpful for designing and planning our application as it gave our team a solid foundation on current WICN listener preferences. The GQP team conducted several surveys that asked current WICN listeners who were signed up for WICN's newsletter, WPI Alumni, and the WPI student body about the different kinds of music applications they use. In addition, these surveys asked how long the respondents have used a smartphone, what operating system their smartphone runs on, and what features would they like to see in a mobile application. This information helped our project team develop the application in such a way that strongly correlated with the report results.

This GQP report showed that most WICN listeners used iOS devices rather than Android devices. Out of all the respondents, 47% used an iPhone, 47% used an iPad, 30% used an Android phone and 10% used an Android tablet. At the start of this project, we were informed that we would be developing a single application for the Android operating system. After reading the data obtained from this report, we were conflicted between developing a feasible application within the allotted time for this project and giving WICN an application that would be most beneficial. Therefore, we attempted to develop an iOS version alongside the Android version, but ended up developing the latter due to the increased difficulty and lack of experience our project group had with developing an application using the Objective C programming language.

2.1.2 GQP #104 - Strategic Marketing Plan for Collaboration Between WICN Public Radio and WPI

This GQP report (Lewis, S., & Song, Z. 2013) did not focus on a mobile application but instead focused on what WICN can do to increase listenership in its programming. The report contained an abundance of data such as categorized demographics including average jazz listener demographics, average WICN listener demographics, and demographics within the WPI student

body. Although this report wasn't as relevant, it did give our project group an insight on WICN's future plans to expand and a great overview as to what the target audience of our mobile application would look like.

2.2 Analysis of Similar Applications

We researched a variety of radio mobile applications in order to compare features and application design. By going through this process, we found common features and design implementations such as play/pause capabilities, navigation menus, swipe capabilities, and varying methods to display mobile-friendly content. We reviewed the following applications:

2.2.1 TuneIn

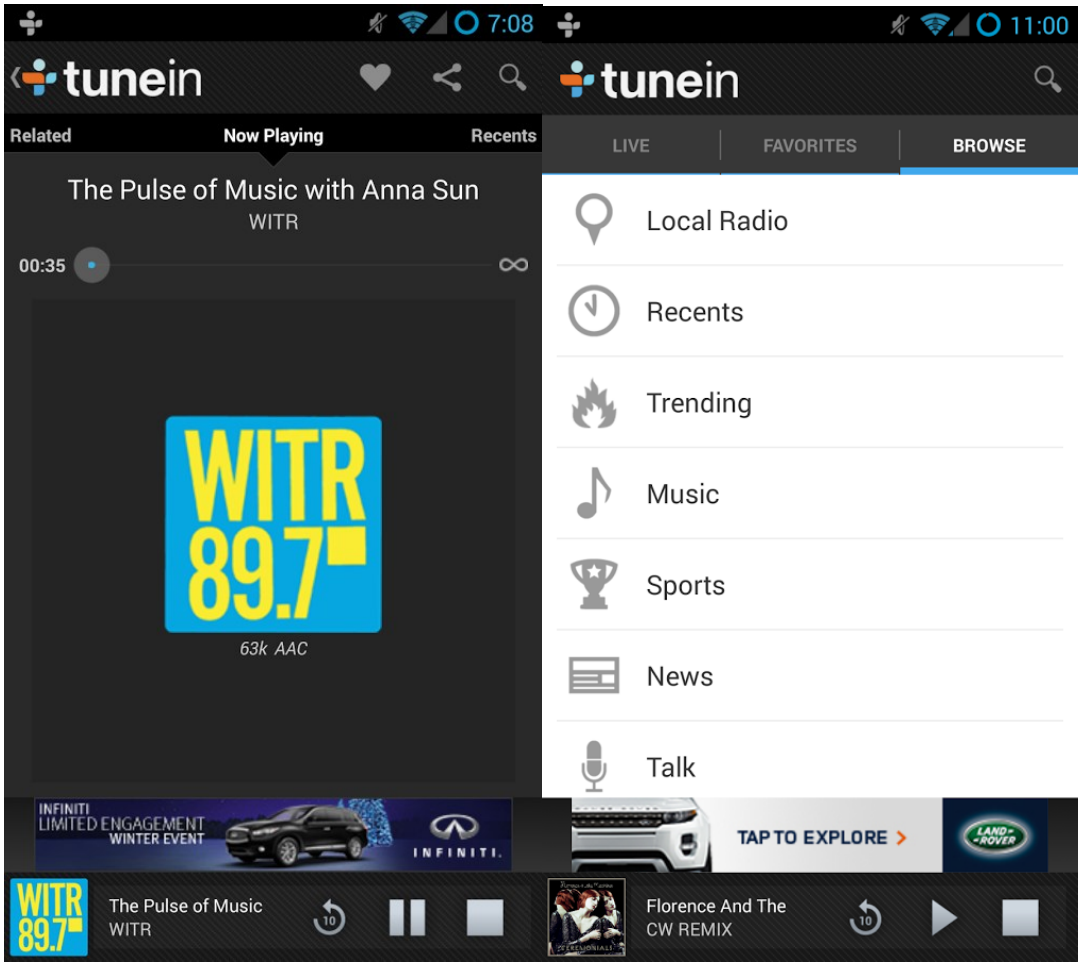


Figure 1: TuneIn Android Application

The TuneIn Android application allows listeners to access over 100,000 radio streams across the globe, including WICN's online stream, and over 2 million podcasts. One of the most useful features of the application is a session management, where the application would save information from the time the listener accessed a radio stream. As shown in Figure 1 above, this

application also includes buttons for playing, pausing, stopping, and even rewinding the stream. This application also has the ability to swipe between categories also shown in Figure 1 such as ‘Now Playing’, ‘Related’, and ‘Recents’ sections.

2.2.2 WayFM

Figure 2: WayFM Android Application

The WayFM Android application has several useful functions that we hoped to implement at the start of this project, some that are relevant to what we were trying to achieve and others that were not. An example of a useful feature that is that this application implements a navigation menu to navigate between different sections. In addition, the application displays information about the current song playing in the stream with the relevant album artwork. In contrast, the application also allows you to select different radio station streams depending on the city and state you selected at runtime as shown above in Figure 2. This feature is outside the scope of our application since WICN has only a single stream that comes from the Worcester area.

2.2.3 US Massachusetts Radio

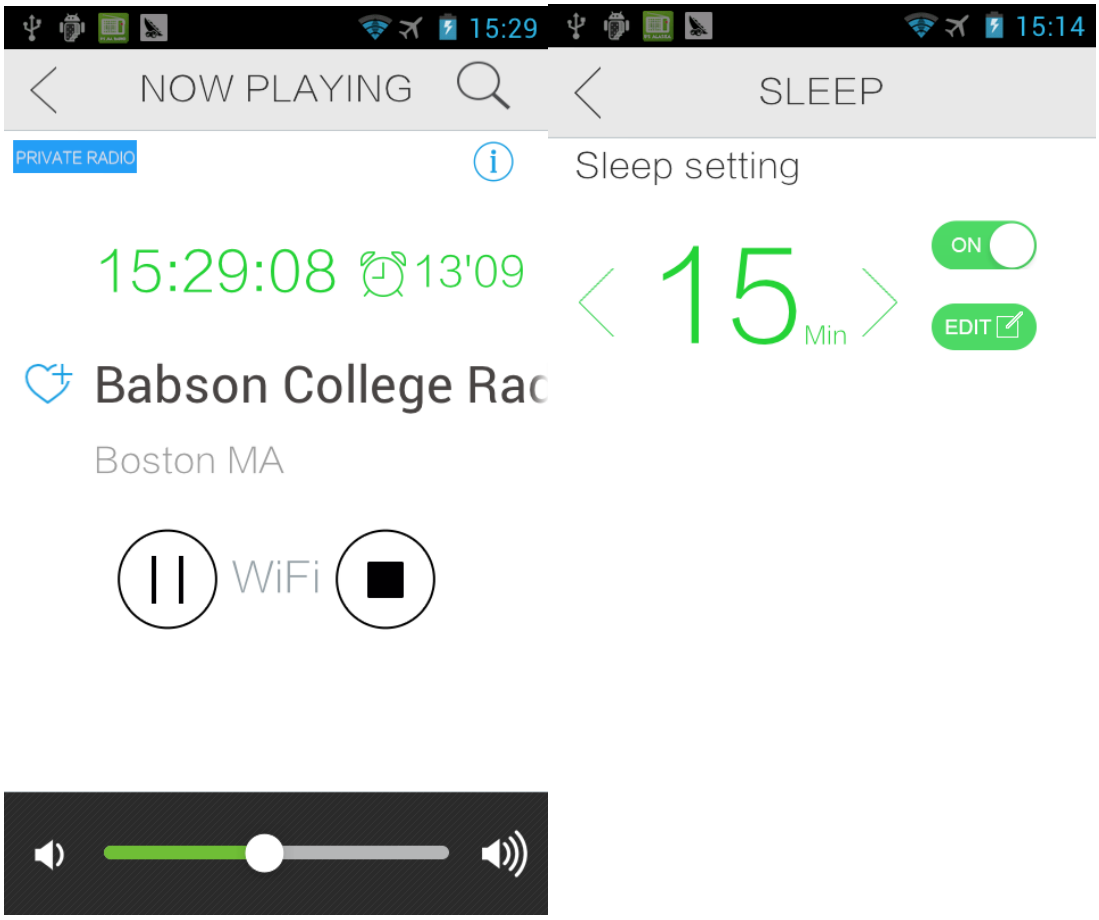


Figure 3: US Massachusetts Radio Android Application

The US Massachusetts Radio Android application included a wide variety of different radio streams available for playback including the WICN online stream. The most prominent feature that we did not like was the design of the user interface elements, as shown in Figure 3 above, and the overall flow between the screens of the application. Navigating between different portions of the application is confusing because nothing is intuitive. A user doesn't know whether to swipe or simply tap on certain user interface elements in order to do something. However, this application did provide some of the functionality that we were planning to implement into this project including a sleep timer to automatically stop the stream after a selected amount of time.

2.2.4 Google Play Music

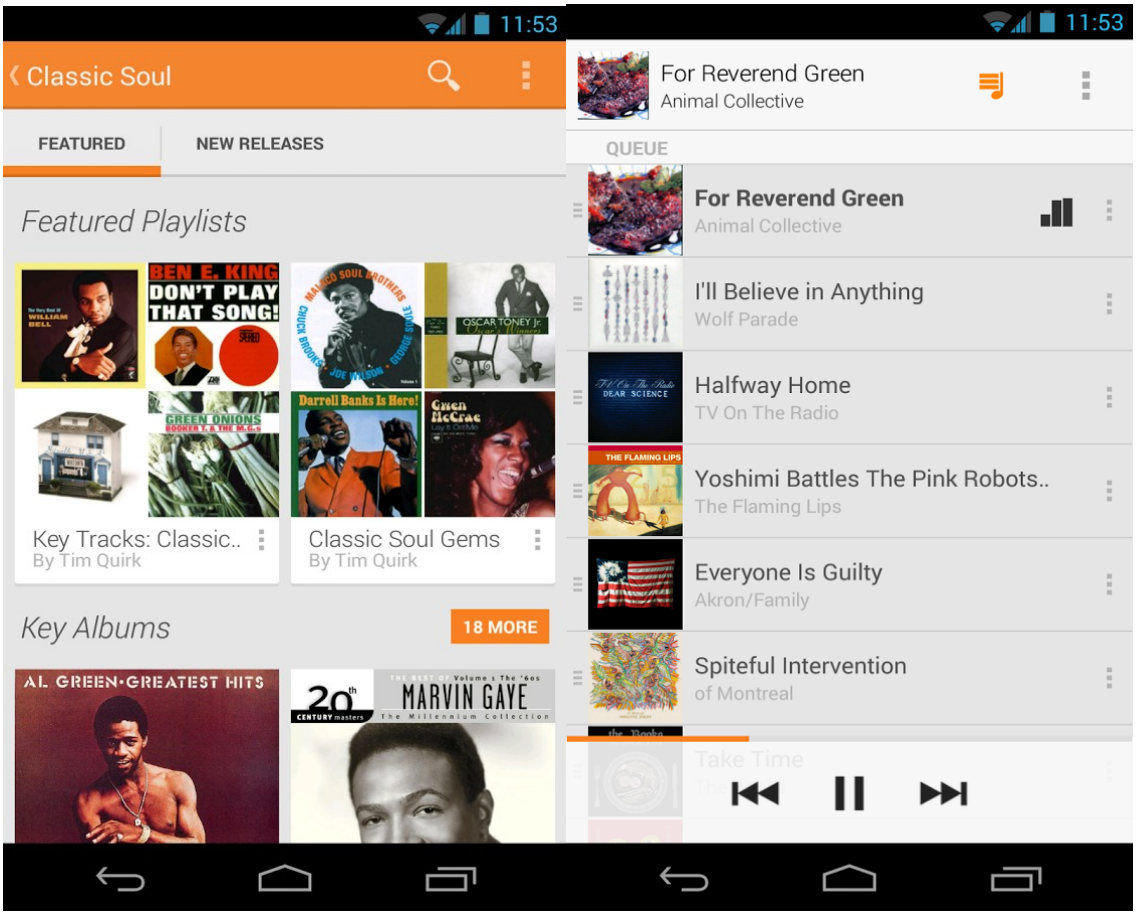


Figure 4: Google Play Music Android Application

The Google Play Music Android application was one of the most useful applications in this reviewing process and it provided us with the most appropriate design that we wanted to implement for our application. Similar to the TuneIn application, it provides the ability to play, pause, and stop the stream from anywhere within the application along with having a slide menu for navigation purposes. In addition, it includes a navigation menu and swipe capabilities for opening and closing the menu and for switching between categorized tabs.

2.2.5 Pandora

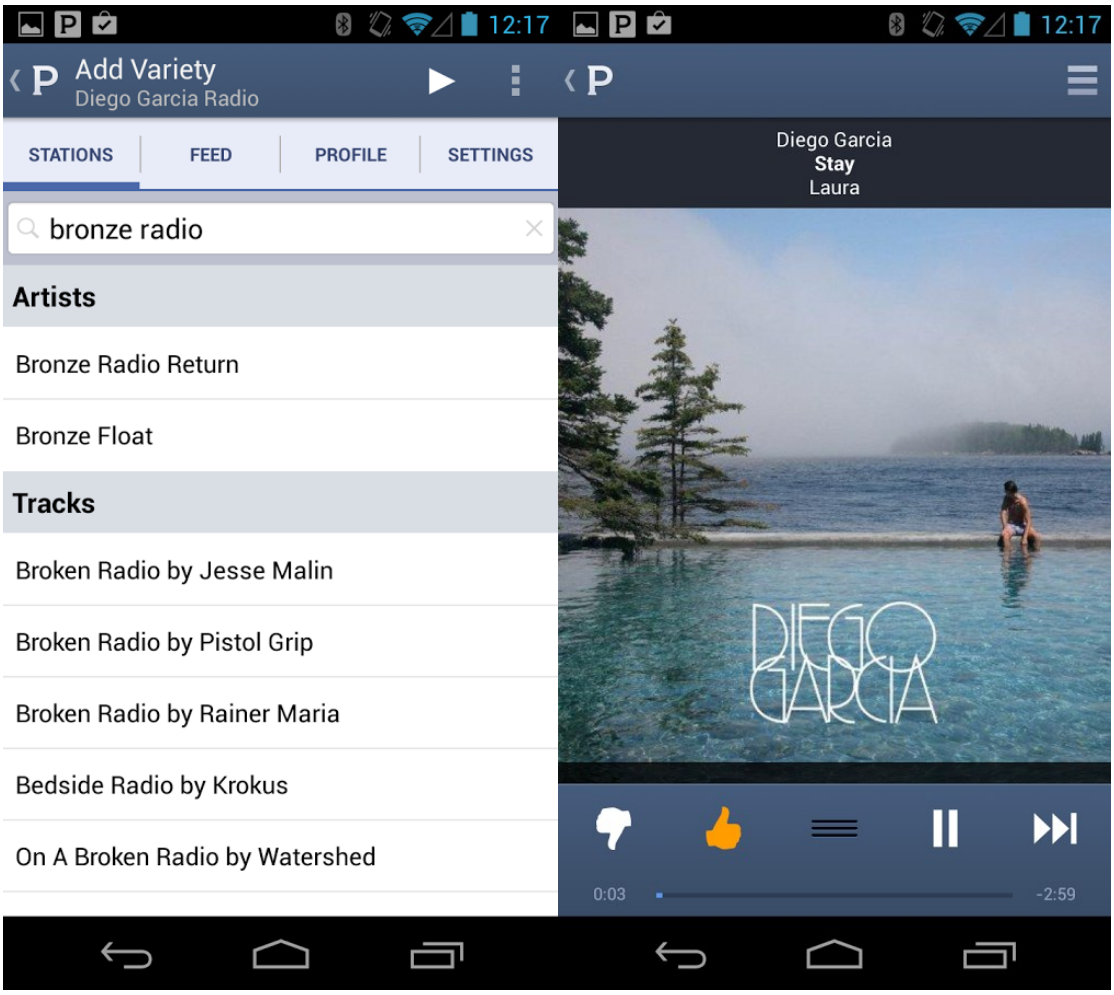


Figure 5: Pandora Android Application

While the Pandora Android application does not offer the capability to listen to live broadcasts, it has one of the cleanest and easiest to use user interfaces of all the applications we studied. Shown in Figure 5 above, the application enables listeners to play or pause music, skip the current song, and adjust the volume from within the application. It also allows the user to view information about the previously listened-to tracks or just the album art of the current one. No matter where the user is within the application, the user is no more than one action away from the 'Now Playing' menu. This application also allows the user to pause or skip the song from the lock screen of the phone: a convenient feature.

2.2.6 iHeartRadio

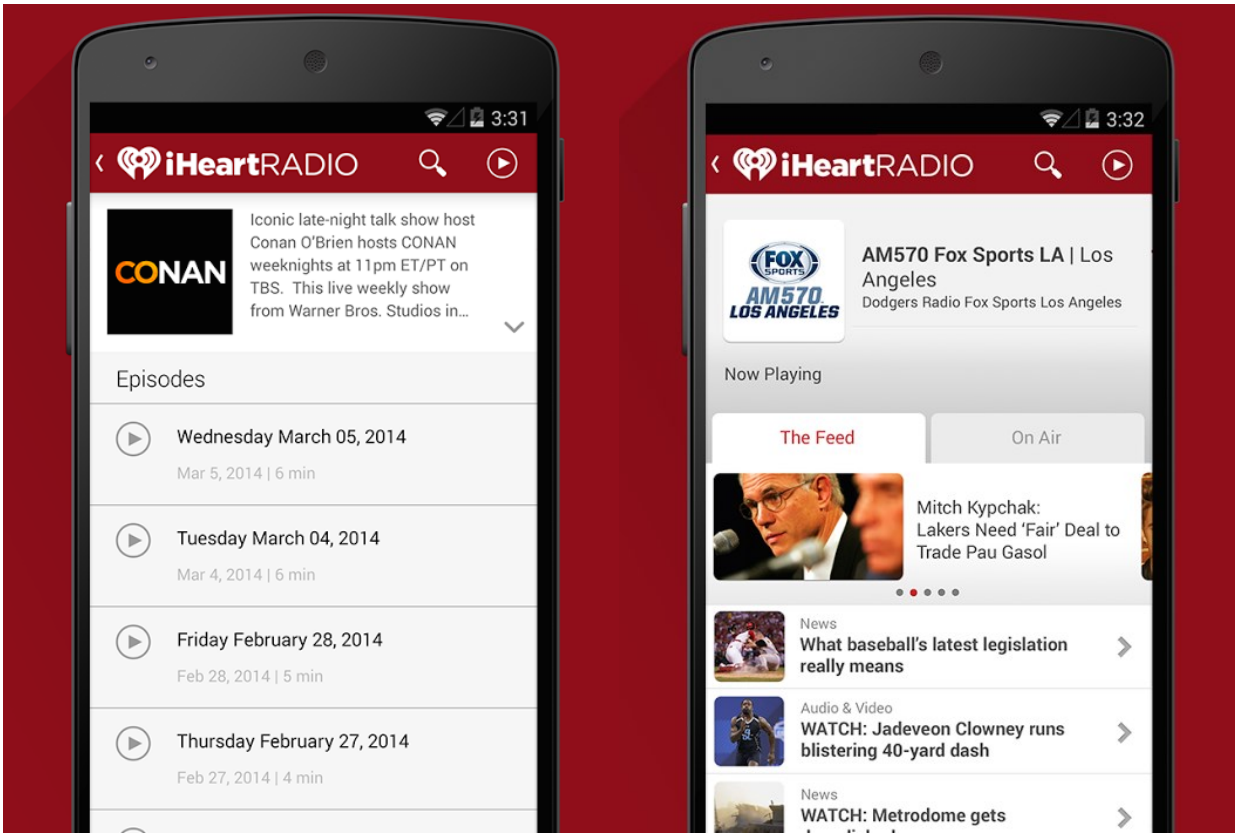


Figure 6: iHeartRadio Android Application

The iHeartRadio Android application is another great example of an application that utilizes online streaming services. Similar to the TuneIn application, users can access a wide variety of radio broadcasts as the application can stream music from every radio station in the iHeartRadio network. The iHeartRadio network mostly consists of at least 800 radio stations in the United States, but also has several international radio stations that can be broadcasted. Similar to the Pandora application reviewed earlier, this application allows users to create custom radio stations by song, artists, or genre of music. As shown in Figure 6, the user interface of this application is much more complicated than the other applications we reviewed with user interface elements such as expandable lists, separate tabs within a screen, and multiple swipe capabilities on a single screen.

2.2.7 WGBH Boston Radio

Figure 7: WGBH iOS Application

We tested the WGBH application on iOS to see if there were any differences in the overall functionality and the designs of the Android applications we tested. Talk about why WGBH is relevant: it is a public radio station in the Boston area. The design of this application is very similar to what our project group wanted to develop including a navigation menu and a donate feature as shown in Figure 7 above along with access to separate podcasts as shown in Figure 8 below.

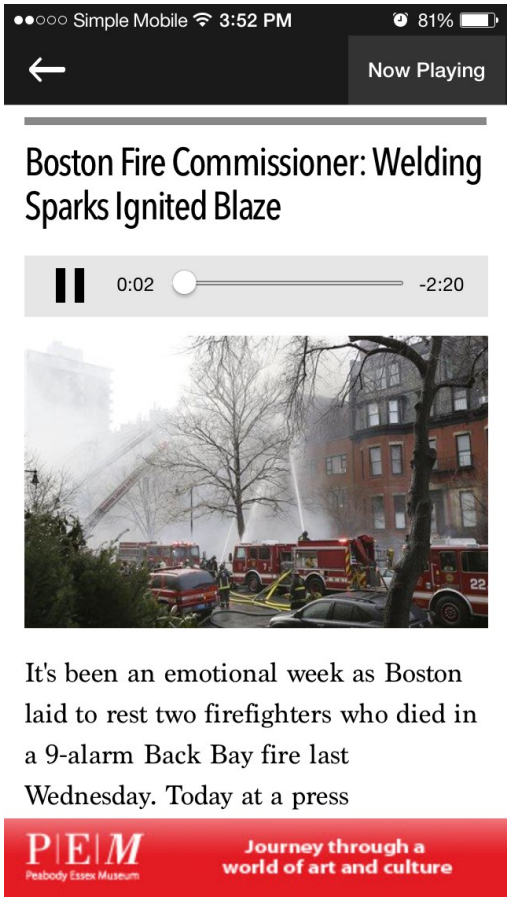


Figure 8: WGBH iOS Application (Podcast)

Unlike the other applications we tested on Android, this particular application only includes buttons to pause and play the stream but no button for stopping it. The version that we tested during the time of this review process seemed to have included a bug that allows a user to listen to the stream and a podcast simultaneously.

2.2.8 KEXP Radio

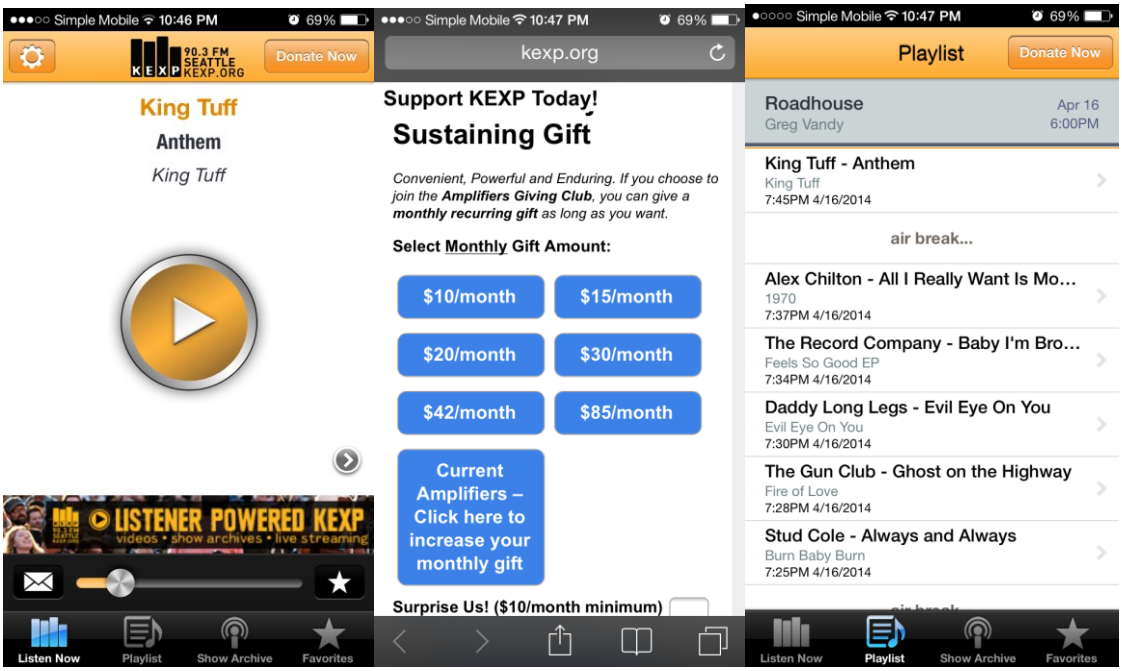


Figure 9: KEXP Radio iOS Application

We looked at the KEXP Radio application as a result of feedback from a WICN staff member who preferred the design over many other applications. Valued features include the radio station's logo at the center of the application header bar along with having a consistent 'Donate Now' button across every portion of the application just as Figure 9 shows. This application also allows users to view what is currently playing in the radio stream and even the ability to select a streaming bit rate of either 64 kbps or 128 kbps.

Chapter 3: Methodology

3.1 Development Platforms

In this project, we developed an Android application for the WICN radio station. Our design was based on our research of similar applications in Chapter 2, the features favored by WICN listeners, and features requested by the staff at WICN.

At the start of this project, we originally decided on developing one application solely for the Android Operating System. This decision was made based on the fact that our team already had experience developing Android applications and already had devices available to test the application on.

As previously mentioned, we addressed the question of simultaneously developing an application for iOS. The aforementioned GQP results revealed that the majority of the station's listeners owned iOS devices. Therefore, we looked further into the complexity of the Android application we were already in the process of developing and noticed that the application was much less complex that we had originally anticipated. Hence, we decided to develop the Android application first. If we had the time, we would begin development on a prototype for iOS. Towards the final weeks of the development process, we discarded the idea completely as we were informed that an iOS version of our application was to be developed by another group in the Fall semester of the 2014-2015 academic year.

3.2 Application Design

Our project group was advised to create a design for our application prior to the start of the development process. Therefore we developed several application mockups using Photoshop CC. In order to keep a consistent color scheme with the current WICN website, our team used the exact fonts, images, and other resources directly from the site itself. We did this so that we would not have to recreate many of the visual elements already provided by the site. Figure 10 shows an example of what we produced:

Figure 10: Initial Android Design for WICN Mobile Application

After presenting these designs to our advisors, we were informed that the designs were not sufficiently branding WICN. For example, the splash screen alone in Figure 10 was not enough to show listeners that this application is from WICN. We were also critiqued on the color scheme and were encouraged to try using different colors for the headers and the backgrounds. Because there were so many design options and implementations, we ended up making several alternative designs that we planned to show the staff at WICN so that they could pick out the one they liked the most.

Figure 11 below shows our overall application flow using screenshots from our first deliverable of our application. Figure 12 shows a much more simplified version of the same diagram using a flow chart. Note that the application starts with a splash screen and immediately goes into the 'Home' screen of our application. Users can then open up the navigation drawer in order to access different sections of our applications as shown below:

Figure 11: WICN Android Application Screenshot Flow

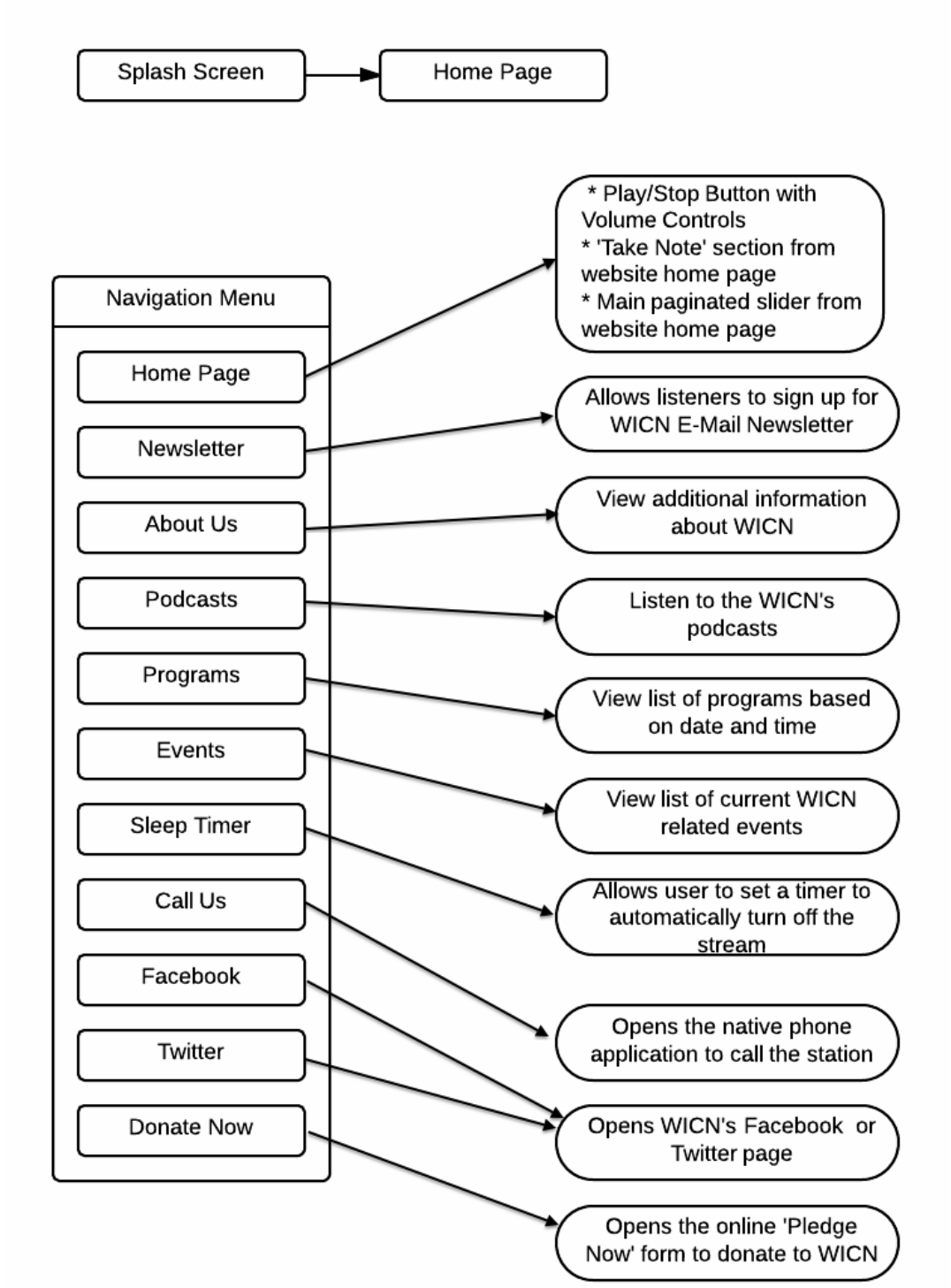


Figure 12: WICN Android Application Modular Diagram

3.3 Current Radio Broadcast Stream

Prior to development, we were curious to see if the station had any additional streams we could access other than the online stream. It turned out that the station only broadcasts one stream at a bitrate of 64 kbps in the mono format. The equipment used to broadcast the HD Radio stream was not functional at the time of this project and required maintenance. This was somewhat discouraging because our group was curious to see if it was possible to access the HD Radio stream in addition to the primary mono stream the station broadcasts. As demonstrated by the KEXP iOS Application, we initially wanted to include a setting for our application that would allow listeners to select the streaming quality of the online radio stream.

WICN currently provides two files to allow other media players such as iTunes, RealPlayer, and more to open them and listen to the online stream. As a result of opening opening these files using a text editor, we were able to access the online broadcast since opening the files revealed the exact URL address used to access the online stream.

3.4 Real Time Track Listing Setback

During the development process, we were informed that the station does not keep track of what current songs are playing at any specific time except for when they run live shows in their Performance Hall. In this case, the host running the live show uses a program called *Spinitron*, which is also directly accessible from WICN website, to keep track of the current song. We were also informed that attempting to retrieve information from this program would be unreliable because not all hosts keep track of the songs they play during these live shows. The ability to display relevant information about the current song playing is a feature that WICN hopes to have in the future.

In addition, we were informed that the station receives an additional broadcast directly from *JazzWorks* from 9AM - 4PM EST, and 11PM - 6AM EST between Monday through Thursday. The entire playlist containing the lineup of songs can also be found directly on the WICN main website. However, there are no timestamps associated with each song entry in the playlist so there is no way to match what song is playing at a certain time. This caused our project group to re-evaluate our project goals and essentially eliminate this real time track listing functionality altogether.

3.5 Android Audio Standards

Since there are many applications on any given Android phone that all want to control the audio for several different reasons it is important for the Android Operating System to have control over the audio system as a whole. Fortunately the Android Operating System already does this using something called an *AudioManager*. The *AudioManager* controls several ‘streams’ of audio depending on what is needed by the application. The one our application was

using was specific for music playback. The application must request that it gets the audio focus from the *AudioManager*. Upon receiving audio focus, an application can then play its audio. The *AudioManager* may also revoke focus from the application at any time if another application requests it, and the application must be able to handle the condition if it's audio focus gets revoked.

We were able to use the *AudioManager* to control the playback of our application for certain cases. For example, if one of the station's listeners is currently listening to the online broadcast and simultaneously receives a phone call, our application uses Android's *AudioManger* to stop the audio playback completely for the duration of the phone call. A similar event handler occurs when they receive e-mails or text messages.

3.6 Adding Drupal Functionality

In order to access features from the WICN website, we had to install, enable, and configure some additional modules on the administration site. This included the 'About Us', 'Events', 'Home', 'Podcasts', and 'Programs' sections of the site. Looking through the documentation listed in the references of this document, we realized that we needed additional credentials to access the FTP server that manages the Drupal administration site for WICN.

Therefore, we installed the *Services* module. This gave us the opportunity to retrieve certain pieces of information directly from the Drupal Content Management System using appropriate RESTful APIs.

With additional research, we figured out that certain aspects of the site were separated into sections. The pieces we had to work with are called *Views* and *Content*. Views in this context are similar to SQL queries in the sense that there is an interaction between a database and how the information is displayed on the site. Views provide some level of abstraction that allows this information to be displayed in some format within the WICN website. The views that our application needed to interact with included the main home page slider along with the 'Events', 'Programs', 'Podcasts', 'On Air Now', and 'Take Note' section from the website. Content in this context are split up into smaller sub-categories such as pages, blocks, polls, articles, etc. The content we needed to access for our application included the 'About Us' page.

We also installed a module called *Services Views* which depends on the *Services* module we previously installed in order to access these Views. After the module was configured, we were successfully able to retrieve the data from the 'Programs' section of the website and the 'Events' section of the website in JSON format. Using these returned values, we programmed our application to call these endpoints upon startup.

3.7 Application Testing

Since the start of the development process, most our application testing was done via our personal Android devices and the built-in Android emulator within our Eclipse ADT bundle. This allowed us to directly test our application from at least two different perspectives and gave us a better account for dealing with different resolutions, speeds, and audio quality.

Towards the end of our development process, we distributed the application to a small group in order to beta-test the application and to collect any sort of bugs and other relevant information from users. Therefore, we developed an application review survey (which was hosted on the WPI Qualtrics server) as shown in the references of this document along with a link to our demonstration application (which was hosted on the Google Play store) and sent this information to the WPI community via appropriate mailing lists.

Chapter 4: Implementation

4.1 Application Code

To develop this Android application, our project group used the provided Eclipse ADT bundle from the Android Developers website. We developed the application using the Java programming language and we used GitHub for source control and bug tracking. With the exception of the Android Support Library (Android Support Library 2014) to increase compatibility with older Android devices and the ViewPagerIndicator project (Wharton, J 2012), we were able to develop this application with no additional third party libraries or external JAR files.

4.2 Application Structure

At the start of this project, we initially thought each part of our application would be programmed as an Android *Activity*, meaning that a brand new screen would get loaded each time a user would tap on a menu item despite whether or not we were reusing the same user interface element. In other words, if our first screen of our application loaded a menu, we would have to 're-draw' that menu again in a separate screen called an *Activity*, rather than reusing what we already added. The sample code from the 'Android Sliding Menu using Navigation Drawer' tutorial in the references used to develop this application suggested that we use something called *Fragments* instead. Fragments in Android are user interface elements that are usually part of an Activity and can be recycled, meaning that they are completely reusable within any part of the application.

This approach turned out to be much more efficient because our application doesn't have to waste additional resources on generating new interface elements that are used over again. In this case, elements that we needed to remain persistent across our application included things such as the navigation menu and the header bar across each part of the application. The 'Design Philosophy' section of the 'Fragments' resource mentioned in the references includes the following figure to explain this in detail:

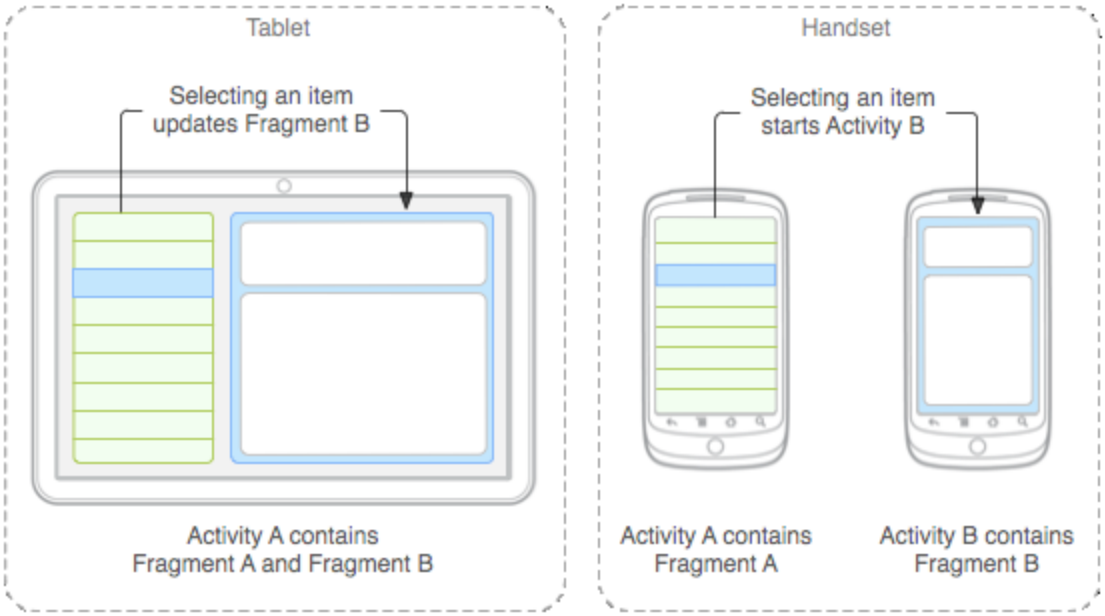


Figure 13: Diagram depiction of how Fragments are used in Android

Based on Figure 13 above, we followed the visual guidelines for a tablet rather than a handheld device. We implemented this using the following snippet of XML code for the layout and the subsequent Java code for the logic:

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <!-- Framelayout to display Fragments -->

    <FrameLayout
        android:id="@+id/frame_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
...
```

```
Fragment fragment = null;
        switch (position) {
        case 0:
            fragment = new HomeFragment();
```

```
            break;
        case 1:
            fragment = new NewsletterFragment();
            break;
        case 2:
            fragment = new AboutUsFragment();
            break;
...
```

The *FrameLayout* in the XML code above is used across each section of our application. This acts as a placeholder for whatever we decide to put there. In this case, we used Fragments to populate this *FrameLayout*. The switch-case shown above is used to connect what section gets loaded based upon what menu item is selected.

4.3 Navigation Menu

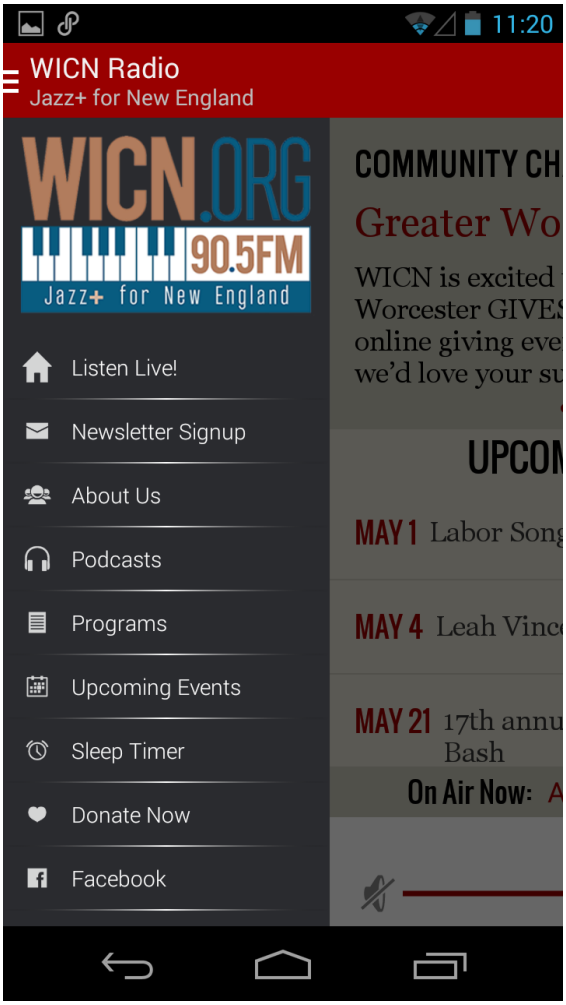


Figure 14: WICN Android Application Navigation Menu

Figure 14 shows the final menu navigation menu that we implemented for our application. Using a combination of resources listed in the references A along with the Android Developer Icon Pack from the Android Developer website, we successfully developed this layout within a week. Most of the development process involved matching our application design and adding the appropriate icons to each section. As previously mentioned, we were able to find most of the icons using the already provided resources from the Android Developer Icon Pack. The rest of the icons were part of either the free sample pack in the Android Icons resource or were purchased from IconFinder and later modified using Photoshop CC according to the Iconography standards set by the Android Developer documentation.

Our design initially included a navigation menu for the purposes of allowing users to switch between one part of the application to another with ease. Based off the applications we reviewed in Chapter 2 of this document, we implemented this because of the popularity of this feature amongst the other applications.

4.4 Threading the Stream

When we first tested loading the stream on the Android application we noticed that while the stream was starting up the user interface would freeze up and become unresponsive. After looking through the Android documentation, we discovered that each every component of an application's process runs in its main user interface thread (Processes and Threads, 2014). This includes *Services* which an application uses to perform work in the background, which we used to play the stream.

In order to create a job that works in a separate thread, you have to explicitly extend the Thread class, instantiate and start the thread. The thread we created RadioThread, for this service contains a MediaPlayer Object and a Handler Object. The Handler Object allows communication between the user interface thread that the service runs on and RadioThread. When the RadioThread Handler receives a Message from the user interface thread, it will call a method of the MediaPlayer, which method it calls is dependent on the contents of the Message. Below is some example code for how the thread works:

```
public RadioThread(){
        mediaPlayer = new MediaPlayer();
        path = streamPath;
         mHandler = new Handler() {
      public void handleMessage(Message msg) {
          if(msg.obj == null){


          }else if(msg.obj.equals(playerEnum.PLAY)){
                try{
                        if(!prepared)
                                setup(path);
                    mediaPlayer.start();
                }  catch (IllegalStateException e) {
                        e.printStackTrace();
                }
        ...
```

The Service containing the RadioThread is the RadioService. Apart from containing the RadioThread, RadioService also acts as the messenger between the user interface thread and RadioThread. RadioService has several public methods that allow the application to control the MediaPlayer in the RadioThread, on the following page is the sample code for the method to stop the media playback.

```
public void stop(){
        radioPlaying = false;
```

```
        Message msg = new Message();
        msg.obj = RadioThread.playerEnum.STOP;
        radioHandler.sendMessage(msg);
    }
```

This method sets an internal variable that monitors the state of the radio to false, so the rest of the application know can know the radio stream is no longer playing. It then creates a message telling the radio to stop and sends that message to the RadioThread. The RadioThread then picks up the message and turns off the radio, disconnecting it from the stream.

The RadioService class was designed to add a layer of abstraction to the implementation of the radio. This abstraction allows the programmer working with the application to not need to know that the a separate thread exists, and that working with the separate thread is just as easy as working with the current thread.

4.5 Fixing Constant Data Retrieval

Figure 15: Consistent Data Retrieval on Android Application

After completing the first iteration of our application, we looked at how much data our

application was retrieving. As shown in Figure 15, we soon figured out that over a period of one week, our application used over one gigabyte of data and eventually realized that the application was not properly destroying the service that controlled the connection to the radio stream. Regardless of whether our application was open and regardless of whether or not the current stream was playing, the service kept obtaining and buffering the stream.

In order to fix this, we implemented several checkpoints from within the application. First, the application checks for an active internet connection at startup. If there is no active connection, a dialog box comes up informing the user that they application cannot continue unless they connect to a mobile or WiFi network. Upon a successful connection, the radio service starts and begins buffering the radio stream. If a user presses the back button on their Android device, another dialog box asks the user whether not they want to close the application. Rather than placing the application in the background, we terminate the application completely if the application is closed.

4.6 Drupal Data Retrieval

After our project group found out exactly how to retrieve the data we needed from the Drupal Content Management System, we needed to implement that data into our existing code. Therefore, we had two options. The first option was to perform a GET request to all the data endpoints we needed at the start of the application. We were already establishing a connection to the online stream at the start of the application, so adding separate processes to get the content we needed at startup proved to be beneficial. That way, everything the application needed to display would already be there and there would be no need to establish additional network connections.

On the other hand, our project group also had the option of establishing these requests whenever a menu item was tapped on. For example, if a user were to tap on the 'Programs' menu item it would make a GET request *just* to that endpoint we established and retrieve the data. The benefit of doing this would be so that we only get what we ask for. The downside was that a new request would have to be made each time a user tapped on the menu item. An example of our code to do this can be found below:

```
JSONParser jsonParser = new JSONParser();
String aboutUsJSON = jsonParser.getJSONFromUrl("http://www.wicn.org/my_services/node/8");
if (aboutUsJSON != null) {
	try {
		JSONObject jObj = new JSONObject(aboutUsJSON).getJSONObject("body");
		JSONArray jArr = jObj.getJSONArray("und");
		for(int i = 0; i < jArr.length(); i++)
		{
```

```
                JSONObject object3 = jArr.getJSONObject(i);
                aboutUsContent = object3.getString("value");

            }

        } catch (JSONException e) {
            e.printStackTrace();
        }
}
```

Chapter 5: Results and Discussion

5.1 Application Survey Results

As previously mentioned, we conducted a survey of WPI students which was hosted by WPI's Qualtrics server. We sent out the survey to several e-mail mailing lists and began collecting data between April 11th, 2014 and April 22nd, 2014.

The most significant information we received from the survey was the qualitative data we asked of the users. The number of respondents to our survey of the application was less than we had anticipated, and under the twenty we had determined to be statistically significant. Since the number of respondents was under the twenty people we determined to be statistically significant, the quantitative data we hoped to analyse is useless. The qualitative data remains useful because its adds a new perspective of what a user might want to see in the application, apart from our team, the developers, and WICN, our clients. The survey also helps to test for any bugs in our software.

One survey respondent using a Droid Razr Maxx running Android 4.4 KitKat reported a problem with playing the music stream. The respondent said:

"I could hear audio, but not until returning to the "Home" screen/activity a second time."
We are not able to replicate this potential bug as we do not have a Droid Razr Maxx to test on and no other survey takers had the same phone. Through Testing we have noticed that sometimes the stream may have some latency in starting audio playback due to a poor network connection. We suspect that this may be the cause of this particular bug.

When we asked if there were any persistent issues the respondents had a few things to say. One respondent said:

"Radio plays for a bit then stops, occasionally resuming. I'm assuming this is just due to a poor connection."

This respondent was in fact correct, as we have noticed the stream start to start and stop seemingly randomly when we had poor network connections on our phones and emulators. A second respondent noticed that:

"Closing the application within the app switching menu stops the music but does not remove the icon from the status bar."

We had not previously found this bug, and were able to replicate it on our own testing

equipment, and have since fixed said bug.

A third respondent noticed that:

"The boxes don't line up well on the newsletter signup page"

We later realized that this particular issue was occurring as a result of the web content formatting. The same formatting issues appeared regardless of whether it was loaded within our application or via a separate web browser.

5.2 WICN Staff Application Review

Towards the final development weeks, our project group sent several screenshots of our application to obtain feedback as to what application features they liked and disliked. We received several responses including requests such as changing the application icon, adding the WICN slogan to each screen, increasing the size of the 'Donate Now' button that appears on every screen, changing the layout of the 'Podcasts' section, and adding an alarm feature in which a user could set an alarm so that the WICN online broadcasts begins to play on a specified time. Our project group spent some time communicating back and forth between the staff in addition to making most of the requested changes for an updated application.

Chapter 6: Conclusion and Future Work

This application challenged our project team to new heights due to the amount of extensive research and new programming aspects each of us took in order to develop it. Overall, this project was much more than a usual 'assigned project' for a programming course since we were given the opportunity to create something from scratch for an outside organization in order to potentially increase both the station's revenue and the station's listener population. The application itself was probably the most commercial-like application that either of the members in our project group has ever developed, and in the end we were very satisfied with the final deliverable that had no known bugs or errors.

6.1 Development Limitations

As previously mentioned in Chapter 4, our project group had to overcome a few obstacles throughout the duration of this project ranging from limitations of our own capabilities to the feasibility of all the application's features. At the start of this project, we wanted to include additional features within the application such as being able to view relevant artist, album, and song information about the current song playing within the live stream. We also wanted to develop a donation feature that would allow a user to both signup for the e-mail newsletter and make donations to the station just by using the native user interface elements provided by the Android Operating System rather than implementing web content. Finally, we had also hoped to include a persistent play, pause, and stop button throughout all the screens of the application so that a user doesn't have to navigate to the very first screen in order to play the online broadcast.

Implementing these features would have taken a significant amount of time to develop which would've caused our final product to be delayed. At least half of these features were not feasible during the time of this project due to the specific ways WICN manages their online donations e-mail newsletters, and online stream. Despite all that, having the opportunity to work directly with the staff at WICN proved to be highly beneficial as we were able to make an application that meets their expectations rather than creating an application on our own terms.

6.2 Application Improvements

As far as improvements, this application could potentially benefit from another major update in the near future. Aside from the features mentioned above, the application could also grant listeners the ability to select specific playback frequencies instead of defaulting to the current 64 kpbs mono broadcast. For example, listeners that have access to a faster internet connection would possibly want to listen to the station's broadcast at a higher quality if they were given that option. Making the application perform more efficiently would also be advantageous, especially for listeners that have slower or older Android devices. Finally, an iOS version of this application would be an advantage for the station as they would have dedicated

applications specific for branding and promoting the station for two major mobile operating systems, thus affecting a larger proportion of their listener population.

References

Albin, J. (2014). Native mobile apps | Drupal.org from https://drupal.org/node/1388470. Accessed February 3rd, 2014

Android Support Library | Android Developers (2014) from http://developer.android.com/tools/support-library/index.html, Version 19.1.0. Accessed April 8th, 2014

AudioManager | Android Developers (2014) from http://developer.android.com/reference/android/media/AudioManager.html. Accessed January 27th, 2014

Beyer, G. (2014). Android Icons | Developer Icons 2 from http://www.androidicons.com/. Accessed March 25th, 2014

Building Web Apps in WebView | Android Developers (2014) from http://developer.android.com/guide/webapps/webview.html. Accessed March 2nd, 2014.

Creating a Navigation Drawer | Android Developers (2014) from http://developer.android.com/training/implementing-navigation/nav-drawer.html. Accessed March 1st, 2014

Diwakar, S. (2014). [Tutorial] [How to] Manually create Android Media Player controls from http://www.sapandiwakar.in/tutorial-how-to-manually-create-android-media-player-controls/. Accessed January 26th, 2014

Diana, N., Yao-Hua Chung, K., & Li, X. (2013). GQP 105 - WICN Mobile App Market Research (pp. 30): Worcester Polytechnic Institute

Fragments | Android Developers (2014) from http://developer.android.com/guide/components/fragments.html. Accessed March 2nd, 2014

FragmentActivity | Android Developers (2014) from http://developer.android.com/reference/android/support/v4/app/FragmentActivity.html. Accessed March 2nd, 2014

Frankenstein, T. (2011). Drupal & PhoneGap - Mobile Application with Drupal 7 Services, PhoneGap & JQuery Mobile for Android - Example from http://tylerfrankenstein.com/code/android-app-with-drupal-7-services-phonegap-and-jquery-

mobile. Accessed February 23rd, 2014

Google Play Music Android Application from https://play.google.com/store/apps/details?id=com.google.android.music, Version 5.4.1409N. Accessed January 23rd, 2014

Hoiste, H. (2010). Quick Tip: Customize Android Fonts - Tuts+ Code Tutorial from http://code.tutsplus.com/tutorials/customize-android-fonts--mobile-1601. Accessed March 18th, 2014

Icon search engine and market place | Iconfinder (2014) from https://www.iconfinder.com/. Accessed March 25th, 2014

Iconography | Android Developers (2014) from http://developer.android.com/design/style/iconography.html

iHeartRadio Android Application from https://play.google.com/store/apps/details?id=com.clearchannel.iheartradio.controller, Version 4.5.1. Accessed January 23rd, 2014

Lewis, S., & Song, Z. (2013). GQP 104 - Strategic Marketing Plan for Collaboration Between WICN Public Radio and WPI (pp. 19): Worcester Polytechnic Institute

KEXP Radio iOS Application from https://itunes.apple.com/us/app/kexp-radio/id342254135?mt=8, Version 2.1.2. Accessed April 8th, 2014

Media Playback | Android Developers (2014) from http://developer.android.com/guide/topics/media/mediaplayer.html. Accessed January 27th, 2014

Media player - Android playing stream m3u using mediaPlayer - Stack Overflow. (2014) from http://stackoverflow.com/questions/6730702/android-playing-stream-m3u-using-mediaplayer. Accessed January 27th, 2014

Metter, K. (2011). Create a Custom-Styled UI Slider (SeekBar) in Android | MokaSocial from http://www.mokasocial.com/2011/02/create-a-custom-styled-ui-slider-seekbar-in-android/

Processes and Threads | Android Developers (2014) http://developer.android.com/guide/components/processes-and-threads.html. Accessed February 14, 2014

Services | Android Developers (2014)
http://developer.android.com/guide/components/services.html. Accessed February 14, 2014

Styling the Action Bar | Android Developers (2014) from
https://developer.android.com/training/basics/actionbar/styling.html. Accessed March 1st, 2014

Tamada, R. (2013). Android Sliding Menu using Navigation Drawer from
http://www.androidhive.info/2013/11/android-sliding-menu-using-navigation-drawer/. Accessed March 1st, 2014

TuneIn Android Application from https://play.google.com/store/apps/details?id=tunein.player. Version 11.3. Accessed January 23rd, 2014

US Massachusetts Radio Android Application from
https://play.google.com/store/apps/details?id=com.prstudio.radio.usmassachusetts, Version 1.0. Accessed January 23rd, 2014

Van Der Linden, J. (2014). Android Holo Colors from http://android-holo-colors.com/. Accessed March 18th, 2014

Vogel, L. (2014). Multi-pane development in Android with Fragments - Tutorial from
http://www.vogella.com/tutorials/AndroidFragments/article.html. Accessed March 2nd, 2014

WayFM Android Application from https://play.google.com/store/apps/details?id=com.wayfm, Version 1.0. Accessed January 23rd, 2014

WBGH iOS Application from https://itunes.apple.com/us/app/wgbh-news-and-culture/id434660601?mt=8, Version 3.2. Accessed February 17th, 2014

Wessman, P. (2014). Services Views | Drupal.org from https://drupal.org/project/services_views. Accessed March 25th, 2014

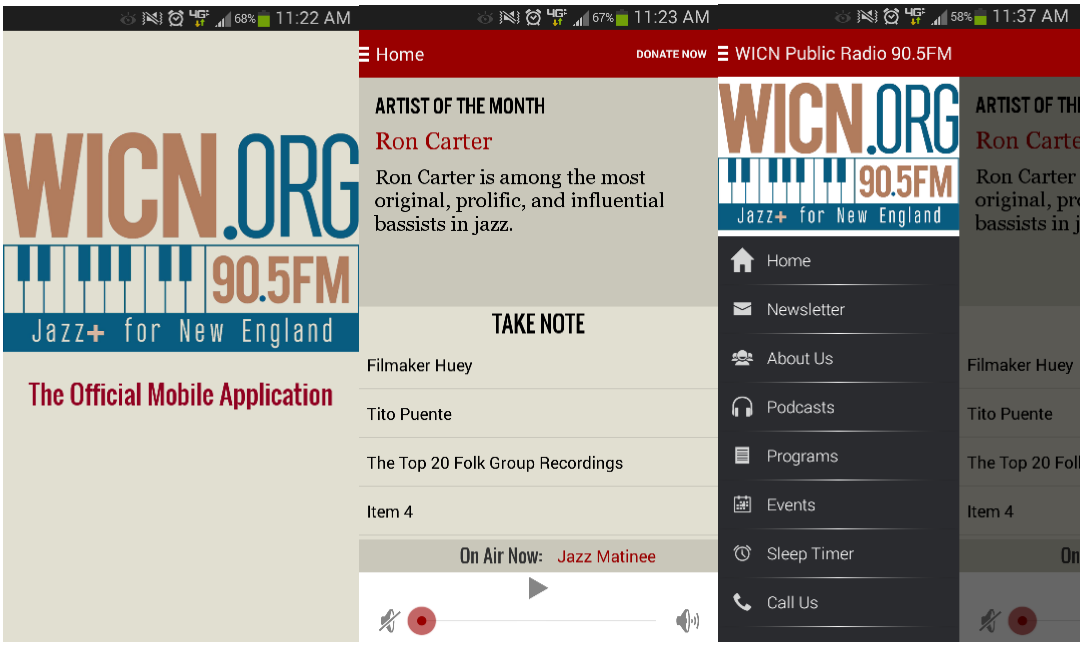
Wharton, J. (2012). ViewPagerIndicator from http://viewpagerindicator.com/, Version 2.4.1. Accessed April 29th, 2014

Appendix A: Application Survey

1. What is your gender?
    - Male
    - Female
2. Please select your age group:
    - Under 18 years old
    - 18 - 20 years old
    - 21 - 22 years old
    - 23-25 years old
    - 26+ years old
3. What Android version are you currently running on your smartphone?
    - Android < 2.0
    - Android 2.0 - 2.1 (Eclair)
    - Android 2.2 (Froyo)
    - Android 2.3 (Gingerbread)
    - Android 3.0 - 3.2 (Honeycomb)
    - Android 4.0 (Ice Cream Sandwich)
    - Android 4.1 - 4.3 (Jelly Bean)
    - Android 4.4 (KitKat)
    - Unsure
4. What smartphone model do you have?
5. How long have you used a smartphone?
    - Less than 6 months
    - 6 months - 1 year
    - 1 - 2 years
    - 2 - 4 years
    - More than 4 years
6. How often do you use applications on your smartphone?
    - Never
    - Less than once a day
    - Once or twice a day
    - Once or twice an hour
    - More than once per hour
7. What kinds of applications do you use in a typical week? (Check all that apply)
    - Entertainment (Games, Videos, eBooks, etc.)
    - Productivity (E-Mail, Calculator, Finance, Word Processing, etc.)
    - Social Networking (Facebook, Twitter, etc.)
    - Educational
    - Personal (Fitness Tracker, Cooking, etc.)

- News (Local, National, International, Weather, etc.)
- Other:

8. If you use your smartphone to listen to music. What kinds of applications do you use to do so? (Check all that apply)
   - Pandora
   - iHeartRadio
   - Spotify
   - Google Play Music
   - Other:
9. Are you currently a listener of WICN?
   - Yes
   - No
10. If your answer to the previous question was "Yes", how long have you been a listener of the station?
    - Less than 6 months
    - 6 months - 1 year
    - 1 - 3 years
    - 3 - 5 years
    - More than 5 years
11. Start up the application. You should see the following screens:



Does the application load successfully with no error messages?

12. If your answer to the previous question was "No", please describe the issue with as much detail as possible.
13. From the Home Screen, press the "Play" button. Does the application begin to play the

online stream? (note it may take a second or two to start playing)
    - Yes
    - No
14. If you answered “No” to the previous question, please describe the issue with as much detail as possible.
15. Open the navigation menu as described in question the third screenshot above. Select any or all of the menu items. Do these pages load successfully with no error messages or crashes?
    - Yes
    - No
16. If you answered “No” to the previous question, please describe the issue with as much detail as possible.
17. While the radio is playing, press the 'Home' button on your android device. Does the stream continue to play while the application in in the background?
18. If you answered “No” to the previous question, please describe the issue with as much detail as possible.
19. Where there any persistent issues that you noticed while using the application?
20. Are there any features of the application that you felt were useless or not necessary? If so, please describe them.
21. Do you have any suggestions for ways to improve the application?
22. Do you have any additional questions? Comments? Compliments?